UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/620,534	07/16/2003	Hoi Chang	P00620-US-00 (19232.0003)	8981

22446 7590 01/03/2006

ICE MILLER
ONE AMERICAN SQUARE
BOX 82001
INDIANAPOLIS, IN 46282-0200

EXAMINER	
KIM, JUNG W	

ART UNIT	PAPER NUMBER
2132	

DATE MAILED: 01/03/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/620,534	CHANG ET AL.	
	Examiner	Art Unit	
	Jung W. Kim	2132	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 December 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *67-100* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *67-100* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on December 5, 2005.

2. Claims 67-100 are pending.

3. Claims 1-66 are canceled.

4. Claims 67-100 are new.

5. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

7. Claims 67-87 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

8. As per claims 67-87, the Specification discloses a general method for adding tamper resistance to software program (fig. 26 and related text); however, it does not

describe an explicit method for adding tamper resistance to a software program using silent guards as recited in these claims. Claims 67-80 define an explicit method that includes steps, "adding a silent guard ...", "selecting a computation in the software program", "determining an expected value of the silent guard variable at the execution point of the selected computation", and "revising the selected computation to be dependent on the runtime value of the silent guard" (see claim 67); "selecting a program variable ...", "determining an expected value of the program variable at a first dependency point the software program", "adding a silent guard variable ...", "determining an expected value of the silent guard variable at the first dependency point", and "making the runtime value of the program variable dependent on the runtime value of the silent guard variable" (see claim 73); "selecting a program variable ...", "selecting a computation in the software program", "determining an expected value of the program variable at the point of execution of the selected computation", and "revising the selected computation to be dependent on the runtime value of the program variable" (see claim 78); "selecting a program block containing a step necessary for proper execution of the software program the program block comprising at least one program instruction", "selecting a silent guard for the program block", "determining the expected value of the silent guard at the start of execution of the program block", and "installing a branch instruction dependent on the silent guard in the software program" (see claim 82). However, none of these methods to add a silent guard are disclosed in the Specification. As stated above, the specification only discloses a general method of adding tamper resistance to a software program (fig. 26 and related text), and the

portions of the specification describing silent guards only refer to silent guards that have already been added to the software programs (pg. 28, line 4-pg. 34, line 10 and pg. 84, line 22-pg. 88, line 6).

9. In addition, the limitation of claim 70 and 79, "selecting a constant value used in the selecting computation; and replacing the constant value with a mathematical expression that is dependent on the runtime value of the [silent guard/program] variable, such that the mathematical expression evaluates to the constant value if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable" are not identified in the specification.

10. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

11. Claim 95 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

12. Claim 95 recites the limitation "the second dependency point" in line 2. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 102

13. Claims 67-69, 71-78, 80-100 are rejected under 35 U.S.C. 102(a) as being anticipated by Collberg "A Taxonomy of Obfuscation Transformations" (hereinafter Collberg).

14. As per claim 67, Collberg discloses a computer implemented method for adding tamper resistance to a software program (pg. 18, section 7.1.3; pg. 19, fig. 18(a-e)), the method comprising:

a. adding a silent guard variable to the software program (pg. 19, fig. 18(a-c), fig. 18(e), the variable "x");

b. selecting a computation in the software program (pg. 19, fig. 18(e), steps 1-10);

c. determining an expected value of the silent guard variable at the execution point of the selected computation (pg. 19, fig. 18(e), steps 5', 7' and 8': assignment of x; and

d. revising the selected computation to be dependent on the runtime value of the silent guard variable, such that the selected computation will evaluate improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable ()pg. 19, fig. 18(e), steps 5', 7' and 8': assignment of c1 and c2.

15. As per claim 68, Collberg further discloses: selecting a program variable in the software program; determining an expected value of the selected program variable at a dependency point in the software program; and making the runtime value of the silent guard variable dependent on the runtime value of the selected program variable at the dependency point (pg. 18, section 7.1.3, especially 6th paragraph; pg. 19, fig. 18(a-e), A, B and C are translated into a1, a2, b1, b2, c1, c2, and x is the runtime value of the silent guard).

16. As per claim 69, Collberg further discloses the step of making the value of the silent guard variable dependent on the runtime value of the selected program variable comprises: computing the runtime value of the silent guard variable using a mathematical expression including the runtime value of the selected program variable and the expected value of the selected program variable at the dependency point (pg. 19, fig. 18(e), steps 5', 7' and 8').

17. As per claim 71, Collberg further discloses the step of revising the selected computation comprises: selecting a computation variable used in the selected computation; determining an expected value of the computation variable at the execution point of the selected computation; and replacing the computation variable with a mathematical expression that is dependent on the runtime value of the silent guard variable, such that the mathematical expression evaluates to the expected value

of the computation variable if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable (pg. 19, fig. 18(e), steps 5', 7' and 8').

18. As per claim 72, Collberg further discloses the step of revising the selected computation comprises: inserting a mathematical expression including the runtime value of the silent guard variable and the expected value of the silent guard variable into the selected computation (pg. 19, fig. 18(e), steps 5'-10')

19. As per claim 73, Collberg discloses a computer implemented method for adding tamper resistance to a software program, the method comprising: selecting a program variable in the software program; determining an expected value of the program variable at a first dependency point in the software program; and adding a silent guard variable to the software program; determining an expected value of the silent guard variable at the first dependency point; and making the runtime value of the program variable dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will equal the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the first dependency point (pg. 19, fig. 18(e), A, B and C is translated into a1, a2, b1, b2, c1, c2 and x is the silent guard variable).

20. As per claim 74, Collberg further discloses the step of making the runtime value of the program variable dependent on the runtime value of the silent guard variable comprises: initializing the program variable to equal the runtime value of the silent guard variable at the first dependency point (pg. 19, fig. 18(e), steps 5', 7' and 8').

21. As per claim 75, Collberg further discloses the step of making the runtime value of the program variable dependent on the runtime value of the silent guard variable comprises: installing a mathematical computation that includes the runtime value of the silent guard variable, such that the result of the mathematical computation is corrupted if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable at the first dependency point; and initializing the program variable to equal the result of the mathematical computation (pg. 19, fig. 18(e), steps 5', 7' and 8').

22. As per claim 76, Collberg further discloses the runtime value of the program variable changes during execution of the software program, comprising: determining an expected value of the program variable at a second dependency point in the software program; and adding a supplementary silent guard variable to the software program; determining an expected value of the supplementary silent guard variable at the second dependency point; and making the runtime value of the program variable dependent on the runtime value of the supplemental silent guard variable, such that the runtime value of the program variable will equal the expected value of the program variable at the

second dependency point if the runtime value of the supplementary silent guard variable equals the expected value of the supplementary silent guard variable at the second dependency point (pg. 19, fig. 18(e), steps 7' and 8').

23. As per claim 77, Collberg further discloses the runtime value of the program variable changes during execution of the software program, comprising: determining an expected value of the program variable at a second dependency point in the software program; and determining an expected value of the silent guard variable at the second dependency point; and making the runtime value of the program variable dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will equal the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the second dependency point (pg. 19, fig. 18(e), steps 7' and 8')

24. As per claim 78, Collberg discloses a computer implemented method for adding tamper resistance to a software program (pg. 18, section 7.1.3; pg. 19, fig. 18(a-e)), the method comprising;

e. selecting a program variable in the software program (pg. 19, fig. 18(e), variables A, B and C);

f. selecting a computation in the software program (pg. 19, fig. 18(e), steps 1-10);

g. determining an expected value of the program variable at the point of execution of the selected computation (pg. 19, fig. 18(a-e), the transformation); and

h. revising the selected computation to be dependent on the runtime value of the program variable, such that the selected computation will evaluate incorrectly if the runtime value of the program variable is not equal to the expected value of the program variable (pg. 19, fig. 18(e), steps 1'-10').

25. As per claim 80, Collberg further discloses the step of revising the selected computation comprises: selecting a computation variable used in the selected computation; determining an expected value of the computation variable at the point of execution of the selected computation; and replacing the computation variable with a mathematical expression that is dependent on the runtime value of the program variable, such that the mathematical expression evaluates to the expected value of the computation variable if the runtime value of the program variable is equal to the expected value of the program variable (pg. 19, fig. 18(e), steps 5'-8').

26. As per claim 81, Collberg further discloses the step of revising the selected computation comprises; inserting a mathematical expression including the runtime value of the program variable and the expected value of the program variable into the selected computation (pg. 19, fig. 18(e), steps 8'-10').

27. As per claim 82, Collberg discloses a computer implemented method for adding tamper resistance to a software program, the method comprising:

i. selecting a program block containing a step necessary for proper execution of the software program, the program block comprising at least one program instruction (pg. 19, fig. 18(e), before transformation);

j. selecting a silent guard for the program block; determining the expected value of the silent guard at the start of execution of the program block (pg. 19, fig. 18(a-d), and steps 1'-8' in fig. 18(e)); and

k. installing a branch instruction dependent on the silent guard in the software program, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the branch instruction will cause an incorrect branch to be taken (pg. 18, section 7.1.3, especially 6th paragraph; pg. 19, fig. 18(e), after transformation, steps 8'-10').

28. As per claim 83, Collberg further discloses the step of selecting a silent guard comprises: adding a silent guard variable to the software program; using the silent guard variable as the silent guard; and installing an initialization instruction for the silent guard variable that executes prior to the branch instruction in the software program, the initialization instruction setting the silent guard variable equal to the expected value of the silent guard (pg. 19, fig. 18(e), steps 1'-7', x is the silent guard variable).

29. As per claim 84, Collberg further discloses the step of selecting a silent guard comprises: selecting a program variable in the software program; and using the program value as the silent guard (pg. 19, fig. 18(e), A, B and C is translated into a1, a2, b1, b2, c1, c2 and x).

30. As per claim 85, Collberg further discloses the step of selecting a silent guard comprises:

l. selecting an insertion point in the software program; selecting a program variable in the software program; determining the expected value of the program variable at the insertion point; making the runtime value of the silent guard dependent on the runtime value of the program variable, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point (pg. 18, section 7.1.3, especially 6th paragraph; pg. 19, fig. 18(a-e), A, B and C are translated into a1, a2, b1, b2, c1, c2, and x is the runtime value of the silent guard).

31. As per claim 86, Collberg further discloses the step of making the runtime value of the silent guard dependent on the runtime value of the program variable, comprises:

m. installing a mathematical computation that includes the runtime value of the program variable, such that the result of the mathematical computation is corrupted if the runtime value of the program variable is not equal to the

expected value of the program variable at the insertion point; and setting the silent guard equal to the result of the mathematical computation (pg. 19, fig. 18(e), steps 5', 7' and 8').

32. As per claim 87, Collberg further discloses the silent guard computation uses both the expected value of the program variable and the runtime value of the program variable (pg. 19, fig. 18(e), steps 8'-10').

33. As per claim 88, Collberg discloses a recordable computer media having a tamper resistant software program recorded there on, comprising: a software program comprising one or more program instructions; a program computation in the software program at an execution point; a silent guard variable in the software program having an expected value at the execution point; wherein the program computation is dependent on the runtime value of the silent guard variable, such that the program computation will evaluate improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable (pg. 18, section 7.1.3, esp. 6th paragraph; pg. 19, fig. 18(a-e), A, B and C are translated into a1, a2, b1, b2, c1, c2, and x is the runtime value of the silent guard).

34. As per claim 89, Collberg further discloses the media further comprising a program variable in the software program having an expected value at a dependency point in the software program, wherein the runtime value of the silent guard variable is

dependent on the runtime value of the program variable at the dependency point (pg. 18, section 7.1.3, especially 6th paragraph; pg. 19, fig. 18(e), steps 5', 7' and 8').

35. As per claim 90, Collberg further discloses the runtime value of the silent guard variable is also dependent on the expected value of the program variable at the dependency point (pg. 19, figs. 18(c), 18(d) and 18(e), steps 5', 7' and 8').

36. As per claim 91, Collberg discloses recordable computer media having a tamper resistant software program recorded thereon, comprising:

n. a software program comprising one or more program instructions; a program variable having an expected value at a first dependency point in the software program; and a silent guard variable having an expected value at the first dependency point; wherein the runtime value of the program variable is dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable equals the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the first dependency point (pg. 18, section 7.1.3, especially 6th paragraph; pg. 19, fig. 18(a-e), A, B and C are translated into a1, a2, b1, b2, c1, c2, and x is the runtime value of the silent guard).

37. As per claim 92, Collberg further discloses the media further comprising an initialization instruction initializing the program variable to equal the runtime value of the silent guard variable at the first dependency point (pg. 19, fig. 18(e), steps 5', 7' and 8').

38. As per claim 93, Collberg further discloses the media further comprising a mathematical computation that includes the runtime value of the silent guard variable, the program variable being dependent on the result of the mathematical expression; wherein the result of the mathematical computation is corrupted if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable at the first dependency point (pg. 19, fig. 18(e), steps 5', 7' and 8').

39. As per claim 94, Collberg further discloses media of claim 91, further comprising a supplementary silent guard variable having an expected value at a second dependency point in the software program; wherein the expected value of the program variable at the second dependency point is not equal to the expected value of the program variable at the first dependency point; and the runtime value of the program variable at the second dependency point is dependent on the runtime value of the supplementary silent guard variable, such that the runtime value of the program variable equals the expected value of the program variable at the second dependency point if the runtime value of the supplementary silent guard variable equals the expected value of the supplementary silent guard variable at the second dependency point (pg. 19, fig. 18(e), steps 4'-8', each of the reassignments to the "C" variable).

40. As per claim 95, Collberg further discloses the expected value of the program variable at the second dependency point is not equal to the expected value of the program variable at the first dependency point; and the silent guard variable has a second expected value at the second dependency point; and the runtime value of the program variable at the second dependency point is dependent on the runtime value of the silent guard variable at the second dependency point, such that the runtime value of the program variable equals the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the second dependency point (pg. 19, fig. 18(e), steps 4'-8', each of the reassignments to the "C" variable).

41. As per claim 96, Collberg discloses a recordable computer media having a tamper resistant software program recorded thereon, comprising: a software program comprising one or more program instructions; a program block containing a step necessary for proper execution of the software program, the program block comprising at least one program instruction; a silent guard in the software program having an expected value at the start of execution of the program block; and a branch instruction in the software program dependent on the silent guard, wherein the branch instruction will cause an incorrect branch to be taken if the runtime value of the silent guard is not equal to the expected value of the silent guard (pg. 18, section 7.1.3, esp. 6th

paragraph; pg. 19, fig. 18(a-e), A, B and C are translated into a1, a2, b1, b2, c1, c2, and x is the runtime value of the silent guard).

42. As per claim 97, Collberg further discloses the silent guard is a silent guard variable, the silent guard variable being set equal to the expected value of the silent guard prior to the branch instruction in the software program (pg. 19, fig. 18(e), steps 5', 7' and 8').

43. As per claim 98, Collberg further discloses the media further comprising a program variable having an expected value at an insertion point; wherein the runtime value of the silent guard is dependent on the runtime value of the program variable, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point (pg. 19, fig. 18(e), program variables are A, B and C).

44. As per claim 99, Collberg further discloses the runtime value of the silent guard is made dependent on the runtime value of the program variable using a mathematical computation that includes the runtime value of the program variable, wherein the result of the mathematical computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point (pg. 19, fig. 18(e), steps 5', 7' and 8').

45. As per claim 100, Collberg further discloses the runtime value of the silent guard is also dependent on the expected value of the program variable at the insertion point (pg. 19, fig. 18(e), steps 5', 7' and 8').

Claim Rejections - 35 USC § 103

46. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

47. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

48. Claims 70 and 79 are rejected under 35 U.S.C. 103(a) as being unpatentable over Collberg.

49. As per claim 70, the rejection of claim 67 under 35 USC 102(b) is incorporated herein. (supra) In addition, Collberg further discloses a method of converting static variables into procedural data as an obfuscation technique. In an example, a constant string within a selected computation is replaced by a function wherein a parameter of the function determines the string generated by the function (pgs. 18-19, sections 7.1.4 and fig. 19). Furthermore, this function is effectively a mathematical expression dependent on the variable parameter and evaluates to the constant string value if the run time value of the variable parameter is equal to the expected value of the variable parameter. Therefore, it would be obvious to one of ordinary skill in the art at the time the invention was made for the step of revising the selected computation to comprise the steps: selecting a constant value used in selecting computation; and replacing the constant value with a mathematical expression that is dependent on the runtime of the silent guard variable, such that the mathematical expression evaluates to the constant value if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable. One would be motivated to incorporate these steps since it prevents unscrupulous users from identifying the operation of the software (Collberg, ibid). The aforementioned cover the limitations of claim 70.

50. As per claim 79, the rejection of claim 78 under 35 USC 102(b) is incorporated herein. (supra) In addition, Collberg further discloses a method of converting static variables into procedural data as an obfuscation technique. In an example, a constant string within a selected computation is replaced by a function wherein a parameter of

the function determines the string generated by the function (pgs. 18-19, sections 7.1.4 and fig. 19). Furthermore, this function is effectively a mathematical expression dependent on the variable parameter and evaluates to the constant string value if the run time value of the variable parameter is equal to the expected value of the variable parameter. Therefore, it would be obvious to one of ordinary skill in the art at the time the invention was made for the step of revising the selected computation to comprise the steps: selecting a constant value used in selecting computation; and replacing the constant value with a mathematical expression that is dependent on the runtime of the program variable, such that the mathematical expression evaluates to the constant value if the runtime value of the program variable is equal to the expected value of the program variable. One would be motivated to incorporate these steps since it prevents unscrupulous users from identifying the operation of the software (Collberg, ibid). The aforementioned cover the limitations of claim 79.

Conclusion

51. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Collberg et al. USPN 6,668,325 discloses an apparatus and method to obfuscate code.

Chow et al. USPN 6,594,761, 6,779,114, 6,842,862 disclose an apparatus and method to tamperproof and obscure computer software code.

52. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Communications Inquiry

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jung W. Kim whose telephone number is 571-272-3804. The examiner can normally be reached on M-F 9:00-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Jung W Kim
Examiner
Art Unit 2132

December 22, 2005

THOMAS R. PEESO
PRIMARY EXAMINER